(12) UK Patent Application (19) GB (11) 2 308 780 (13) A

(43) Date of A Publication 02.07.1997

(21) Application No 9526615.1

(22) Date of Filing 28.12.1995

(71) Applicant(s)
**Nokia Telecommunications OY**

**(Incorporated in Finland)**

**Mäkkylän puistotie 1, FIN-02600 Espoo, Finland**

(72) Inventor(s)
**Colin Grant**
**Matthew Faupel**
**Graham French**
**David Bending**
**Phil Heighes**

(74) Agent and/or Address for Service
**Frank B Dehn & Co**
**179 Queen Victoria Street, LONDON, EC4V 4EL, United Kingdom**

(51) INT CL$^6$
**H04Q 3/00**

(52) UK CL (Edition O)
**H4K KF42**

(56) Documents Cited
**EP 0553560 A2**

(58) Field of Search
UK CL (Edition O) **G4A AUDB , H4K KF42**
INT CL$^6$ **G06F 9/46 , H04Q 3/00**
**ONLINE : WPI**

(54) **Telecommunications network management**

(57) A telecommunications network management system has a database DB including information about the network TN, the information being in the form of objects that relate to the network elements NE to be managed by one or more operatives running applications APPLN at one or more parts PP1 ... PPN peripheral to a core part CP. The core part CP stores all objects used by the system as master copies in database DB. A peripheral part has a cache CH for separately storing copies of objects used by its applications APPLN1 ... APPLN, and also has information (eg. a table TBL1) indicating which applications use which objects. The core part CP also stores information (eg. a table TBL2) indicating which peripheral parts have copies of which objects. The objects may be subject to changes due to events in the network TN or due to running an application APPLN, and the core and peripheral parts communicate with one another to keep the objects in the core part and the copies in the peripheral parts consistent with one another. A copy of an object is removed from a peripheral part when it is no longer needed by any of the applications in that peripheral part. More than one operative may work at the workstation WS of a peripheral part, in which case the objects required by each operative's applications can be stored as copies in respective caches CH, and copies of objects required by the applications of all the operatives at a peripheral part may be stored in a workstation cache (WCH, Fig.6).

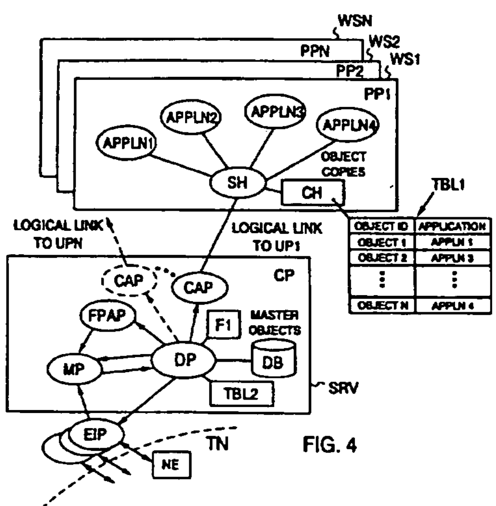


FIG. 4

GB 2 308 780 A

This print takes account of replacement documents submitted after the date of filing to enable the application to comply with the formal requirements of the Patents Rules 1995

BEST AVAILABLE COPY

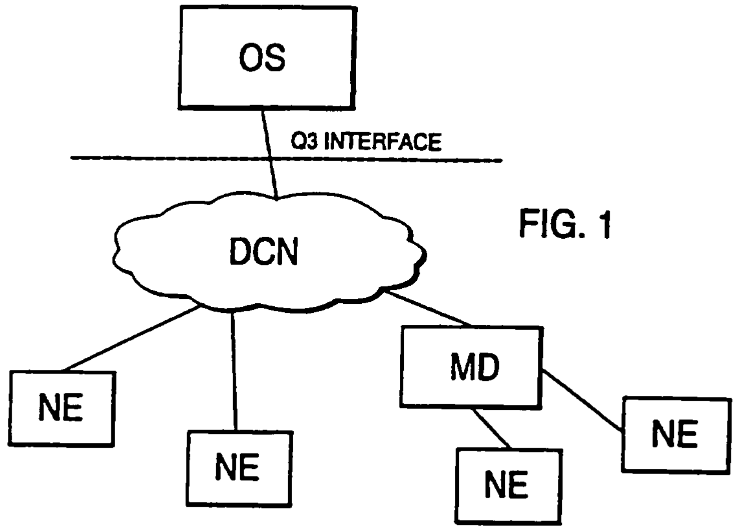
OS
Q3 INTERFACE
DCN
NE
NE
MD
NE
NE

FIG. 1

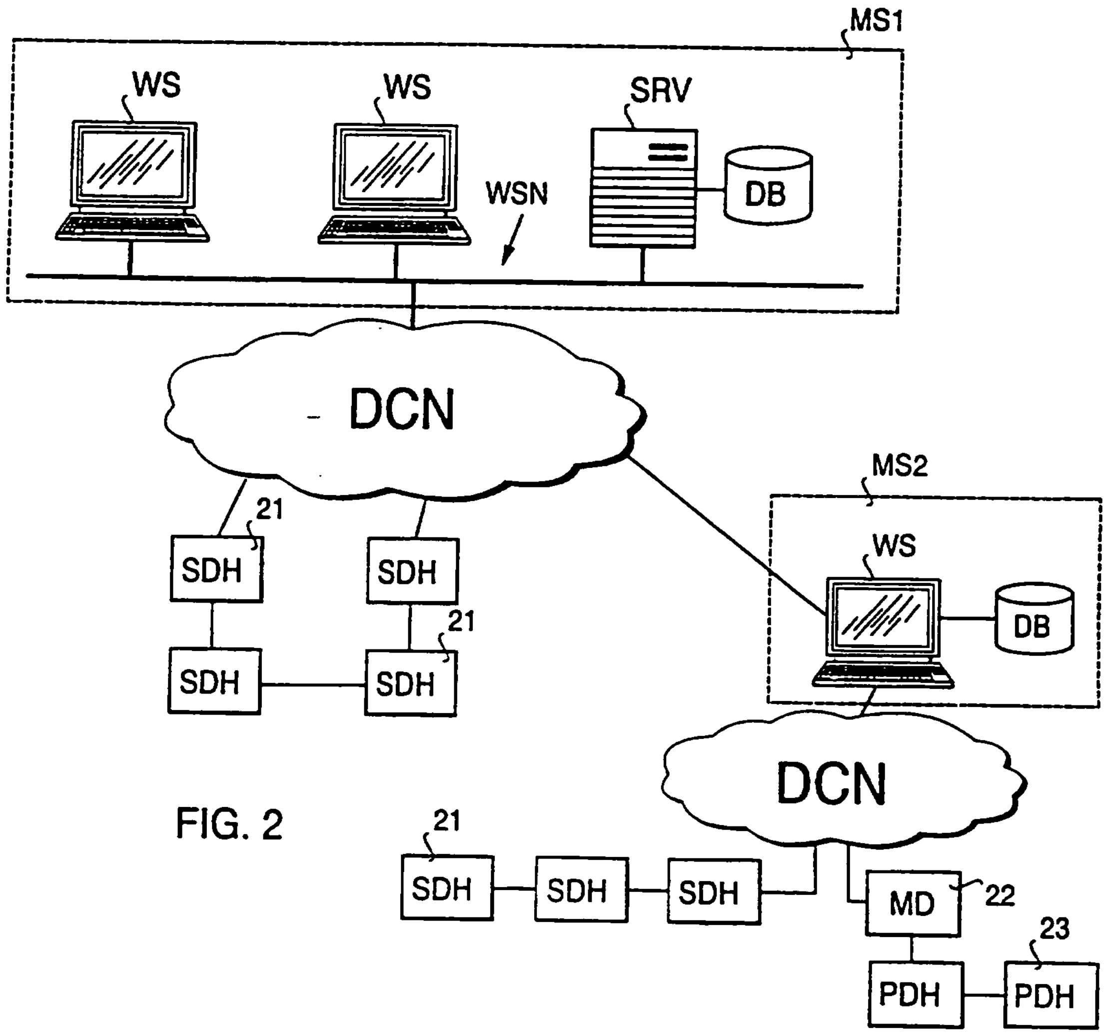
MS1
WS
WS
SRV
DB
WSN
DCN
21
SDH
SDH
21
SDH
SDH
MS2
WS
DB
DCN
21
SDH
SDH
SDH
MD
22
23
PDH
PDH

FIG. 2

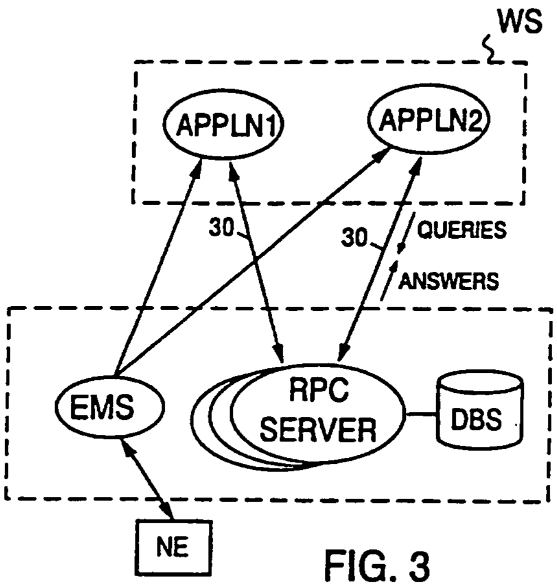
WS
APPLN1
APPLN2
30
30
QUERIES
ANSWERS
EMS
RPC SERVER
DBS
NE

FIG. 3

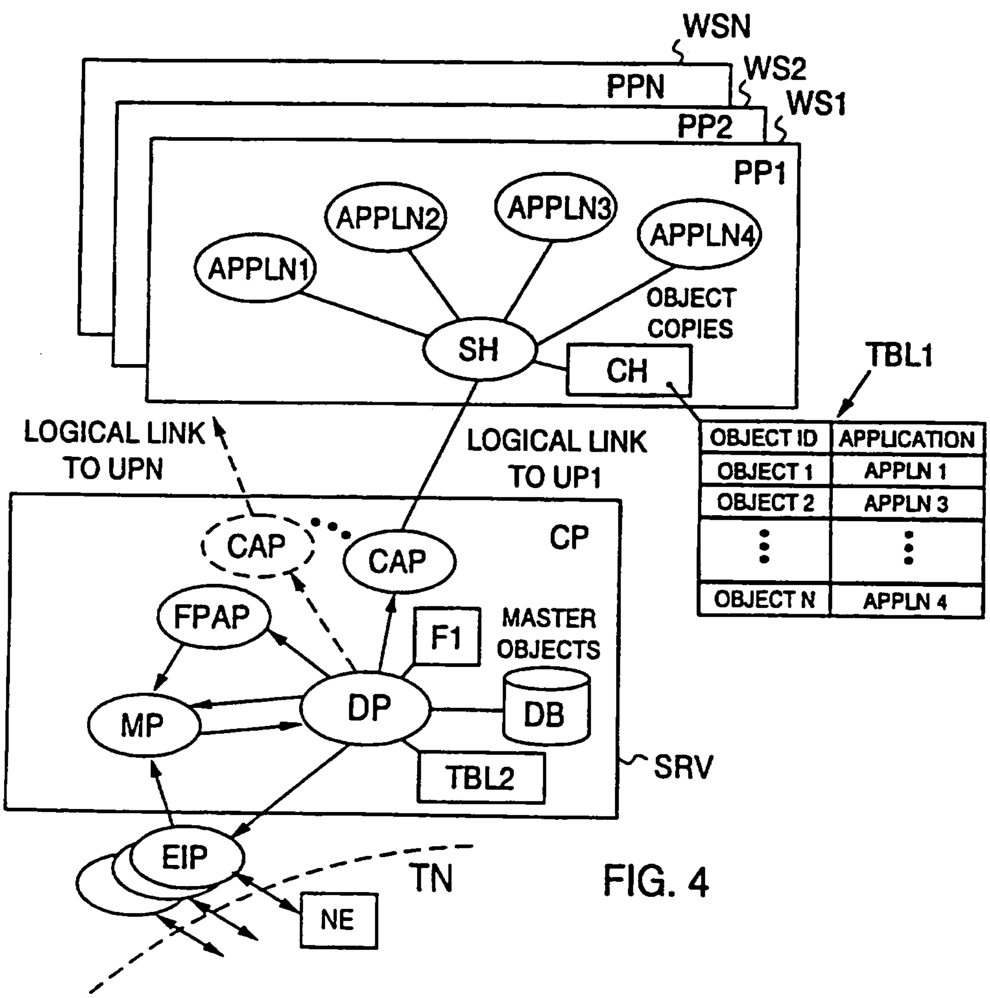
WSN
WS2
WS1
PPN
PP2
PP1
APPLN1
APPLN2
APPLN3
APPLN4
OBJECT COPIES
SH
CH
TBL1
OBJECT ID
APPLICATION
OBJECT 1
APPLN 1
OBJECT 2
APPLN 3
OBJECT N
APPLN 4
LOGICAL LINK TO UPN
LOGICAL LINK TO UP1
CAP
CAP
CP
FPAP
F1
MASTER OBJECTS
MP
DP
DB
TBL2
SRV
EIP
TN
NE

FIG. 4

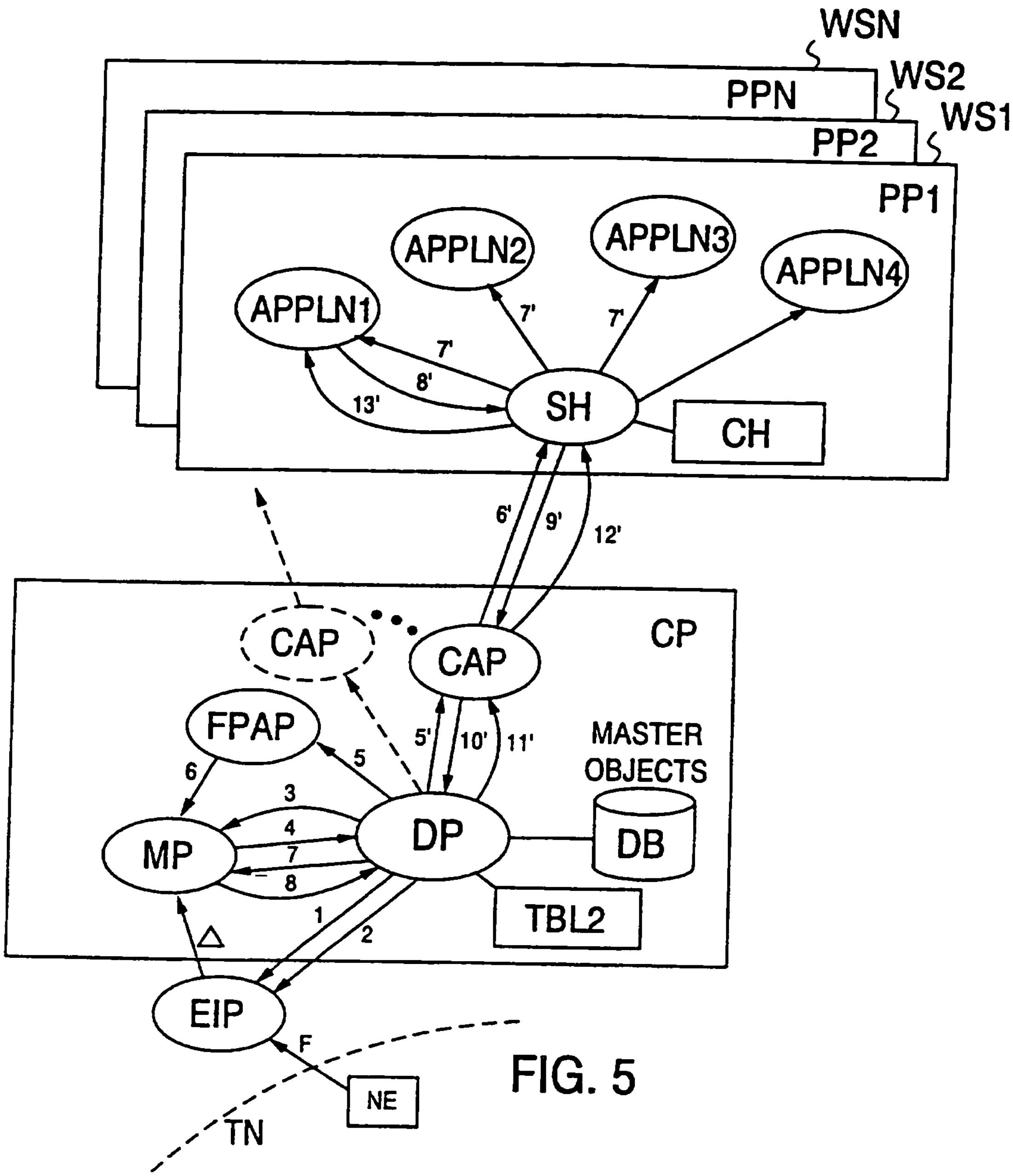
WSN
WS2
WS1
PPN
PP2
PP1
APPLN1
APPLN2
APPLN3
APPLN4
7'
7'
7'
8'
13'
SH
CH
6'
9'
12'
CP
CAP
CAP
FPAP
5'
10'
11'
MASTER OBJECTS
5
6
3
4
7
8
MP
DP
DB
TBL2
1
2
EIP
F
NE
TN

FIG. 5

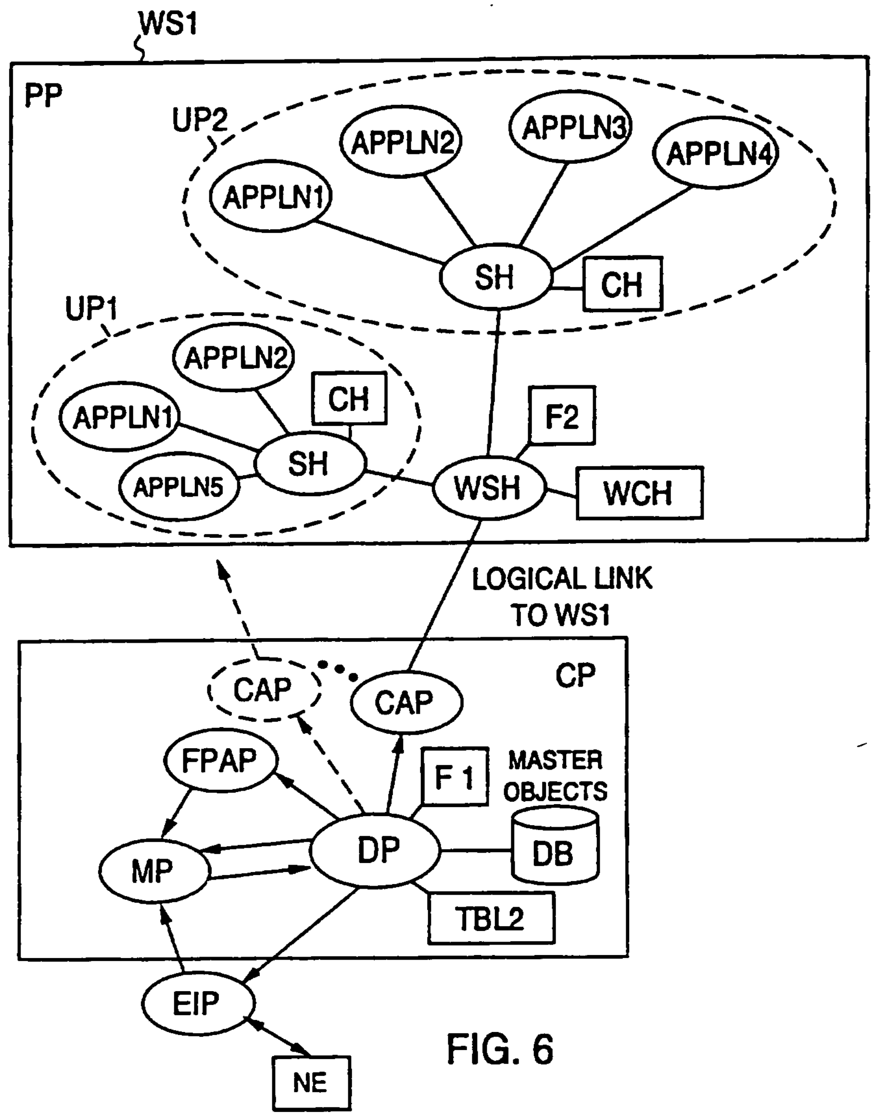
WS1
PP
UP2
APPLN1
APPLN2
APPLN3
APPLN4
SH
CH
UP1
APPLN2
CH
APPLN1
SH
APPLN5
F2
WSH
WCH
LOGICAL LINK
TO WS1
CAP
CAP
CP
FPAP
F 1
MASTER
OBJECTS
MP
DP
DB
TBL2
EIP
NE

FIG. 6

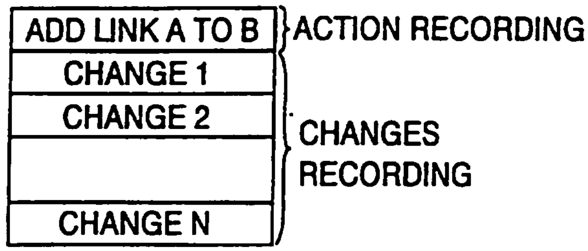
ADD LINK A TO B
ACTION RECORDING
CHANGE 1
CHANGE 2
CHANGES
RECORDING
CHANGE N

FIG. 8

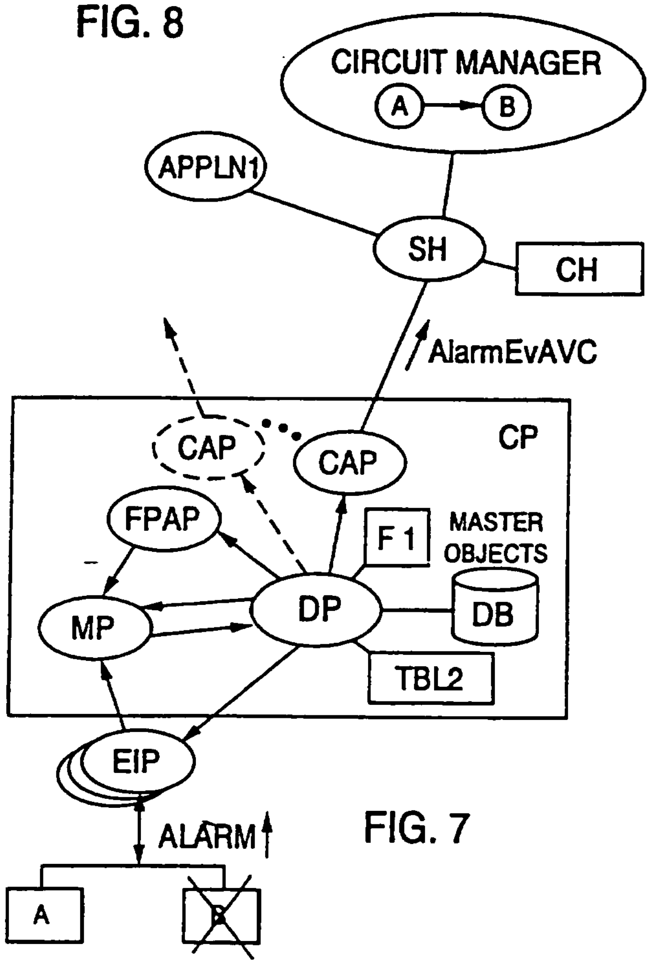
CIRCUIT MANAGER
A
B
APPLN1
SH
CH
AlarmEvAVC
CP
CAP
CAP
FPAP
F 1
MASTER
OBJECTS
DB
MP
DP
TBL2
EIP
ALARM
A
B

FIG. 7

2308780



Telecommunications network management system

The present invention relates to a management system according to the preamble of attached claim 1 for managing a telecommunications network. The telecommunications network to be managed may be e.g. a SDH (Synchronous Digital Hierarchy) network, a PDH (Plesiochronous Digital Hierarchy) network, or a combination of such networks.

The basic situation in network management is usually such that an operator managing a telecommunications network, e.g. a telephone company, has a plurality of customers (i.e. network users) in addition to the physical network. The operator sells the customers various services that utilize the network. (A public network will be used herein as an example; in principle, however, the same description applies to a private operator managing e.g. an organization network). To meet customers' data transmission requirements in the physical network, the operator utilizes a number of facilities or operative processes for the provision of customer services. These operative processes can be divided into groups in accordance with the functions for which they are intended:

- Service Provisioning taking care of the performance of customer services, including e.g. invoicing customers for services.

- Operation & Maintenance for keeping the network operative to allow the usage of customer services. One of the most important functions in this respect is the supervision and repair of network faults.

- Planning & Development, the function of which is to develop network operation so as to better meet customers' needs and to increase the overall profit-

ability of the operator enterprise.

As appears from the above, network management takes place on several different levels, depending on the extent to which the functions to be performed on a specific level are associated with the overall management of the operator enterprise. The management of a telecommunications network is generally divided into four different levels, which are from bottom to top as follows:

- network element management layer,
- network management layer,
- service management layer, and
- business management layer.

This division is used e.g. in the ITU-T (the former CCITT) recommendation M.3010, which specifies a kind of framework for the management architecture of a telecommunications network. The bottom layer below the above four layers is the equipment itself; these equipments are managed by installation and field engineering tools.

The network element management layer means the management of an individual network element (such as a multiplexer or a cross-connection device) as a separate component without simultaneously paying attention to the condition of the network or other network elements. The majority of so called "network management" systems commercially available today are actually network element management systems within this layer.

The network management layer is concerned with the management of the entire telecommunications network, such as overall management of network connections. One example is the creation of connections and the end-to-end supervision of their condition. This means that e.g. alarms detected on equipment are

not just displayed against that equipment, but they are also propagated to show what services (paths and circuits) are affected by the fault, if any. The present invention is positioned in this layer.

As distinct from the above, the service management layer is not concerned with technical network management. It takes care of e.g. customer data, supervision of services provided to customers, invoicing for services, and considering needs for services of different types.

The business management layer is used to monitor and plan the business activities and economy of the entire enterprise, resulting in decisions affecting the lower levels.

At present, network management systems are changing into systems that manage the telecommunications network as a whole, whereas conventional management systems have handled only the remote control of transmission equipment, especially monitoring alarms produced by the equipment. In conventional network management methods, configuration changes, such as creation of new end-to-end connections, have been laborious and time-consuming, as the end result consists of several configuration events the prerequisite of which is that the maintenance staff of the network first gets an overall view of the situation and then decides on configuration changes required in individual network elements. In new network management systems, on the contrary, an overall view of the network and its condition is produced within the system, and the system itself gives the required configuration commands to each transmission equipment. As a consequence, all configuration changes can be performed significantly more rapidly than previously. Such developments have been accelerated by the freeing of

competition in the field of telecommunications.

The above-mentioned recommendation M.3010 specifies the management architecture as shown in Figure 1. The architecture basically consists of one or more operations systems OS connected to a data communication network DCN communicating with an actual telecommunications network which is to be managed and which includes the network elements NE managed. It is to be noted that the connections of the data communications network and those of the telecommunications network are logically distinct, although they can be implemented physically in one and the same cable. Logically, there are thus two networks: (a) a network providing services to customers, and (b) a network maintaining the service provisioning network. The management of certain transmission equipments (network elements) further requires a separate Mediation Device MD, which mainly acts as a protocol converter between a Q3 interface complying with the recommendations and transmission equipments that do not understand the protocol applied in the interface but use their own proprietary protocol. New SDH equipment, for instance, can be connected directly to the Q3 interface, whereas older PDH equipment requires a Mediation Device.

In practice, a management network for a combined SDH and PDH network may be e.g. such as shown in Figure 2. Users (network operator staff) sitting at the operation centre use network management work stations WS connected to a separate local area network WSN, which may be e.g. an Ethernet network. The management system is typically distributed in several computers of the local area network, one of the computers being a dedicated server machine SRV having a database DB containing information necessary for

managing the network. In its practical embodiment, the local area network further comprises e.g. necessary back-up devices (like DAT drives or mirrored disks) and event-logging printers (not shown).

The management system is connected via the above-mentioned Q3 interface e.g. to the SDH network. A variety of alternatives have been defined for the Q3 interface, so that the interface may be e.g. an X.25-type packet-switched interface or an Ethernet LAN interface. (The packet-switched interface is useful if the operator in charge of the network management also otherwise uses a packet-switched network.) In practice, control channels between the SDH network elements 21 are established in the overhead bytes of the STM-N signal (N = 1, 4, 16), so that control signals between SDH equipments propagate with the payload signal (that is, also in the same physical network). Such control channels established in the overhead bytes are called Embedded Control Channels, and they are formed e.g. in the STM-1 frame by the section overhead bytes D1 to D12.

PDH equipments, on the contrary, need manufacturer-specific solutions, wherefore they have to be connected to the management system through a separate mediation device 22.

The management system may also be hierarchical so that different geographical areas have their own smaller management systems that together form an integral management network. For instance, a management system covering one country may be divided geographically into smaller management systems operating in different parts of the country. Each smaller management system takes care of the management of the network portion in the concerned geographical area. In the example of Figure 2, management systems MS1 and

MS2 geographically apart from each other form together a single common management system and management network.

Network management standards are nowadays largely based on so-called object-oriented descriptions, though the standards do not require the use of this technique. Objects are data structures in a network management system, which describe the functions and state of a network component. An object is thus an element having certain attributes ("I am like this") and certain operations ("I can do these things"). (In the object-oriented approach, objects with the same data structure (attributes) and the same behaviour (operations) are grouped into a class. A specific implementation of an operation is called a method and each object is said to be an instance of its class.) A typical object is e.g. a cross-connection device with certain attributes (cross-connections that are active) and certain methods (e.g. make cross-connection and release cross-connection).

In a telecommunications network management system, objects can be physical ones or logical ones. Physical objects are elements that form part of the physical network. Such objects are e.g. the above-mentioned network elements (a network element is any piece of telecommunication equipment that works as a single unit to provide telecommunication functions) or physical connections (such as optical fibres, twisted pair cables, coaxial cables, radio links or satellite links). Logical objects are logical entities that do not form a single piece of the physical network. Such objects are e.g. paths and circuits. (A path is a connection of a fixed bit rate and format between two physical interfaces within the network. A circuit is a connection set up for a customer, between two physical

interfaces on the boundary of the network. Thus, a circuit usually comprises several consecutive paths.)

A network object may have a number of different attributes. Some attributes (such as "fault state") are used by several different types of object. In addition, for some types of network object (such as a route), it is convenient to define an attribute which consists of a collection of other attributes. Typical attributes are e.g. "availability status", "fault state" and "operational state". The attributes have different possible values, e.g. fault state can have values:

- OK. There are no problems.

- Warning. There are outstanding faults, but these do not affect services.

- Degraded. Some or all of the services provided by the object are degraded.

- Failed. All the services provided by the object are lost.

- Unknown. The fault state of the object is unknown.

The "operational state", in turn, can have e.g. two different values:

- Enabled. The object can operate, either completely or in part.

- Disabled. The object cannot operate at all.

As obvious from above, a network operator's biggest asset is the transmission system as such, and the management system is a tool for running efficiently a business based on the services provided by the transmission system. Therefore, the management system should be able to follow the dynamic use of the network at every moment to allow the operator to make decisions that help the operator to make the best profit in all circumstances. For such decisions the

operator might for example want to know what should be fixed at this moment, how much spare capacity there is in the network, how the individual lines are utilized, where the restriction in the bandwidth is, etc. If the network management system cannot give the operator the information needed to understand the use of the network at every moment, the operator cannot really control the network and run the business in the best possible way.

Most management systems are based on RPCs (RPC, Remote Procedure Call). The architecture of such a system is shown in Figure 3. The management system has typically a central machine called an RPC server. This RPC server has objects representing the various network elements stored inside it. The application processes (as will also be disclosed later, an application is a functional system block containing the user interface functionality, i.e. allowing the user to configure the network and view network events on the screen of the workstation) are stored in a workstation WS and they include local objects called RPC stubs which relate to the network elements to be managed. When an application wants to have information on a specific object, the corresponding stub object sends a question to the remote RPC server via a link 30 between them. The RPC server then retrieves the information from the object in question, packages the result, and sends it back to the application that made the question. If the application wants to get e.g. seven pieces of information to show them on the screen of the workstation, it would have to send seven messages to the RPC server and wait for seven replies. The local application itself does not know that messages are transmitted to the RPC server. It is, of course, possible to send seven queries or answers at

the same time, but also in this case problems will occur if the application wants e.g. 8 or 9 pieces of information, as the structure and coding of the messages has to be changed.

The RPC-based systems have a central event management system EMS which broadcasts the events to all applications. If a network element NE, for instance, informs of a failure, the EMS broadcasts information about the event to all applications.

This kind of a system which is based on a centralized database on the RPC server works well for small systems in a normal situation. However, problems arise if one or more applications repeatedly require more information from the RPC server (for example, if there is a search going on). This can lead to a huge queue of events on the RPC server. This in turn leads, at least momentarily, to very slow applications as an individual application has to queue for every piece of information.

The performance of the above-described known management system is particularly problematic in emergency situations (such as a Network Element failure). In the case of emergency, all of the applications receive a fault message from the network and after that all of them request at the same time for details from the RPC server. So, when there is a major fault in the network (for example an STM-1 link is lost), the applications need a lot of information to work out the effects of the failure. It is precisely at this point where the system is suddenly overloaded and the performance of the system drops drastically. Thus, the performance of the system changes in this situation in the wrong way, as the network management system should operate in the most efficient way when there is a fault in the network.

The problem associated with the performance can be illustrated by the following example. In practice, an RPC-based system can typically handle about 100 events in one second (if there is e.g. an Ethernet LAN between the workstations and the RPC server). For example, if an STM-1 link is lost in the network to be managed, the EMS will receive about 2,000 events (an STM-1 link has about 2,000 lower-level channels). As every application is informed about the events, it may take as long as 20 seconds before information about of the fault reaches the application that needs it most. The situation is even worse if the link between the workstation and the RPC server is slower, such as a X.25 connection.

An RPC-based system may also comprise more than one RPC server. In this case, objects are spread among these servers. This, however, does not improve the performance decisively, as a specific object is nevertheless stored in a single server, and the applications have to compete with each other to gain information about the object in question.

The object of the present invention is to provide a novel network management system which does not have the above-mentioned drawbacks, that is, to provide a network management system which allows dynamic control of the network without degrading in failure situations while taking into account the operator's needs. This is achieved by means of the system according to the invention, which is characterized by what is disclosed in the characterizing portion of attached claim 1.

The idea of the present invention is to keep up a full image of the network to be managed in the system core as well as required portions of this full image outside the core in association with the

applications, and thus divide the functionality of the system in a new way. Required images are produced by copying the object data kept up in the core into required locations outside the core. What object copies are needed depends on what kind of applications are used, that is, on the copies needed by the applications.

The system according to the invention remains responsive in all circumstances in the best possible way, i.e. even when faults are detected in the network.

In the following, the invention and its preferred embodiments will be described in greater detail with reference to the example of Figures 4 to 8 of the attached drawings, in which

Figure 1 illustrates a telecommunications network management architecture;

Figure 2 shows an example of a management network for a combined SDH and PDH network,

Figure 3 schematically illustrates the functional architecture of a known network management system;

Figure 4 illustrates the functional architecture of the management system according to the present invention in its basic form;

Figure 5 illustrates the function of the system according to Figure 4 when a fault is received from the network to be managed;

Figure 6 illustrates a preferred functional architecture of the management system according to the present invention;

Figures 7 and 8 illustrate the process used in the system to keep the local copies consistent with the master copies, and vice versa.

The functional architecture of the management

system according to the present invention is illustrated in Figure 4. A system with this kind of architecture is able to operate e.g. in the system shown in Figure 2. The system comprises a single core part CP and one or more peripheral parts PP1...PPN, which are connected to the core part of the system. In this example, each peripheral part is intended for one user (operator employee) of the system.

The system typically has a dedicated core computer, such as a server SRV (also shown in Figure 2). The core part of the system keeps up an updated model of the state and operation of the telecommunications network TN to be managed, of which a single network element is shown in the figure. As described earlier, the core part uses objects to keep up this model. The core computer SRV is connected to one or several workstations WS1...WSN each including one peripheral part. The functional processes of the system, which run on the core computer and on the workstations, have been marked with ovals. These processes will be described in the following.

An application (indicated with the references APPLN1...APPLN4 in Figure 4) is a functional system block which allows the user to configure the network and view network events on the screen of the workstation. The application is thus a system part offering the system's services to the user. An individual workstation is therefore also called an application server.

A Session Handler SH is a functional system part that handles a single user's session with the system and forwards messages between the core and the user's applications. The Session Handler can be associated with the initial system log-on window through which a user logs on and establishes a session with the core.

The Session Handler starts up various applications according to commands given by the user. When the user logs out, the Session Handler performs the necessary closing-down and deregistering operations to terminate the session (the term registering will be explained later).

A Core Access Process CAP verifies that the Session Handler belongs to a known user on a known host. Having done so, it acts as the gateway to the system core. For each CAP in the core part there is a corresponding Session Handler in the peripheral part.

A Database Process DP controls access to the database DB to store persistent information (master objects of the management system). The main functionality of this process is:

- to convert persistent objects to and from a storable format,

- to store the current state of all persistent objects in the database DB and to keep that database up to date by the use of change and event messages sent to it by a Modelling Process MP of the core part,

- to handle the forwarding of event information to interested parts of the system.

The Modelling Process MP is responsible for maintaining the currency of the core's model of the network to be managed and dealing with changes to it. The main functions of the Modelling Process are:

- to accept change and event indications from either EIP or CAP (i.e. either from the network or from the user) and validate them,

- to apply changes to the model if they are valid and to determine the results of these changes,

- to pass the changes to an appropriate External Interface Process EIP, if the changes require corresponding changes in the network to be managed,

- to generate events based on the changes and events received (e.g. fault events on paths and circuits).

The External Interface Process EIP is a functional system part that converts data from the external world (data from the network to be managed) into the internal world of the management system. As far as the system user is concerned, he or she just sees a network element NE, like a multiplexer or a cross-connect device, without having to know the manufacturer or the version of the device. Different types of External Interface Processes are used for different classes of equipment. The main functionality of each EIP is:

- to monitor the network for events and to exchange notifications with the network,

- to translate these events and notifications into equivalent events that are applied to the Modelling Process MP,

- to pass these events and notifications to the Modelling Process,

- to accept changes received from the Modelling Process; translate them into the format of the network model; and send them to the network and then inform the Modelling Process when all commands relating to each original change have succeeded, failed or timed out.

In fact, the Interface Processes could also be included in the core part, but Figure 4 shows them separate, as they form the interface of the management system with the external world. In practice, communication with the external world can take place e.g. through a 7-layer OSI (Open Systems Interface) protocol stack defined for the Q3 protocol. The interface processes are described in the Applicant's

parallel application GB-A-XXXXX, which is referred to for a more detailed description.

A Fault Propagation Agent Process FPAP is a functional system part that works out the actual impact of a received event in the system. In order to be able to do this the FPAP uses information about the object class in question. This information it receives from the Database Process DP.

As the network elements cannot provide all the information needed by the applications, the database DB must store information e.g. about individual network elements and interconnections and interrelations between the network elements, the operations the network elements are capable of performing, and the services provided by the network elements. This information is stored in the form of an object model representing the transmission network TN in terms of objects and their attributes and methods performed on these attributes. Thus, an object is the representation, in the object model, of one of the resources to be managed.

Applications use these objects that form the image of the network to be managed within the system. Therefore, they have to get objects from the core part of the system. An application can also create a new core object to database DB, e.g. as a result of a user command.

Applications also register to receive various events. The registration is passed to the Session Handler and further to the core part of the system. The registration is stored in the form of a filter F1 in association with the Database Process DP. This filter finds out on the basis of the event type, to which Core Access Processes and to which applications a certain type of event is to be transmitted (i.e.

what users are interested in a certain type of event). Events from the core of the system are duplicated in the Session Handler and sent to all applications which have registered to receive them. In this way, all of the user's applications can receive the event at the same time, so that information presented to the user by the different applications is consistent.

In this way the core part of the system knows to which application a certain event has to be transmitted. This also allows unwanted events to be filtered out as soon as possible (in the core) and reduces the bandwidth used between applications and the core part of the system. The core part is thus able to filter events up to the application level. It is also possible to use filters in steps so that the core part knows only up to the level of the Session Handler what event has to be transmitted to the peripheral part. The Session Handler thereafter has a new filter, which indicates in which event a specific application inside the concerned Session Handler is interested. The first alternative, however, is to be preferred in that it does not make the peripheral part (Session Handler) too complicated.

The use of filters is described in the Applicants's parallel patent application GB-A-XXXXX, which is referred to for a more detailed description.

According to the present invention, part of the objects stored in the core part of the management system are placed in a cache CH at each Session Handler SH. (Although a cache is a certain kind of memory, in this connection caching means only that a copy of something is kept to get fast access to it.) Thus, the Session Handler has a copy of any object an application has fetched from the core part of the system. The Session Handler thus also stores object-

specific information indicating which one of its applications uses a specific object. This information can be stored e.g. in the form of a table TBL1 shown in Figure 4. The core part correspondingly knows which Session Handler has a copy of a specific master object. This information can be stored in the form of a table TBL2 in association with the Database Process.

At the system start-up, the Session Handler has no copies of objects. When the application needs a copy for the first time, the copy is fetched from the core part. As the transmitted retrieve message includes information specifying the object to be retrieved and the application that needs the copy, the Session Handler is able to keep up a table (TBL1) indicating which application uses a specific object, and the Database Process is able to keep up a table indicating which Session Handler uses a copy of a specific master object.

If an application fetches an object of which the Session Handler already has a copy, the application receives a copy of the object from the Session Handler without having to interact with the core part of the system. When the application ceases using an object, it transmits a message as an indication of this to the core part through the Session Handler, as a result of which the copy is removed from the memory of the Session Handler and the tables are updated accordingly. If the object is needed again, the copy is fetched from the core in the same way as at the system start-up.

When changes are made to an object in an application, these changes are transmitted through the Session Handler to the core, which reapplies the changes to the master objects to keep the state of the master copies consistent with the state of the copies

in the peripheral part. Once the changes have been accepted and validated by the MP and stored in the DB they are passed onto all other interested Session Handlers and applications so that their copies of the objects are updated.

In the following the operation of the system according to the invention will be described more fully with reference to Figure 5 and using an alarm from the network as an example. For the sake of clarity, filter F1 and table TBL1 are not shown in Figure 5.

At first, a fault occurs in a network element NE of the network to be managed (a single network element is shown in the figure). The network element thus sends an alarm message F concerning the fault to the respective EIP. The EIP first of all retrieves the object that represents the particular network element from the database (arrow 1) and applies the change to that network element. Depending on the location of the fault, it may also be necessary for the EIP to retrieve objects associated with the first object (arrow 2). The fault indication may come e.g. from a 4 x 2 Mbit/s multiplexer, but the EIP finds out on the basis of the fault data and the multiplexer object it has retrieved that the actual fault is e.g. in one of its 2-Mbit/s input interfaces. The EIP thus finds out on the basis of the information it receives from the network which one of the system objects is concerned (e.g. the above-mentioned interface object) and what the fault is like.

The EIP then sends a notification of the object and the change occurred in it to the Modelling Process. This message is indicated with the reference Δ, which indicates that only information about the change that has occurred in the object is sent to the

Modelling Process.

The Modelling Process thereafter gets a copy of that object (arrow 3) from the database, applies the alarm to it, and sends it back as a delta (Δ) change message (arrow 4), which describes the changes of that (interface) object. At this stage the change has thus been updated in a master copy of the concerned object in the database DB. The fault code F received from the network to be managed can also indicate more than one alarm, which are then similarly updated in the database. According to a preferred embodiment of the invention, the Modelling Process can also implement caching of objects so that frequently changed objects do not need to be repeatedly fetched from the database DB.

The Core Access Process CAP and the Fault Propagation Agent Process FPAP have previously registered to the Database Process DP to inform the Database Process what type of events they are interested in receiving. As one of those types is a new fault, the delta change is then copied to the FPAP process (arrow 5).

The purpose of the FPAP process is, as mentioned above, to work out the actual impact of the fault within the management system. If e.g. the 2-Mbit/s interface is connected to a 2-Mbit/s path object which has e.g. up to 31 times 64 kbit/s circuit objects, the FPAP process will send a report (arrow 6) to the Modelling Process telling the Modelling Process MP to change the state of all these objects to be "failed". On the basis of the received information the Modelling Process retrieves all these objects from the database (arrow 7), changes them, and stores them again in the database DB (arrow 8). All of the related master objects in the database DB have now been updated.

At the same time as the delta change is copied into the FPAP, it is also copied into access processes CAP (arrow 5') that have registered to the Database Process as processes that want to receive an event of that type. Accordingly, the delta change is sent once from the Core Access Process in question to the corresponding Session Handler (arrow 6'). It may then be distributed n times between the applications (arrows 7').

Consequently, the Database Process has sent one message to the FPAP and one message to each CAP that wants to receive any event of that type. In the Session Handler the received message has been copied n times into all interested applications. Depending on the application in question, some of them (like application 1 in Figure 5) may then respond to the received message by asking for more information (arrow 8'). This message is then transmitted to the Database Process (arrows 9' and 10') and a reply is sent back (arrows 11', 12' and 13').

For some of the applications, the delta change (arrow 7') may not be sufficient but the applications may also need the object to get more information on the actual impact of the delta change, for example. The applications are event-based. This means that when they receive an event (delta change), they act according to predetermined rules, like "get the object, process the object this way, present it on the screen of the workstation, apply it to the printer, and put it into a log".

It is further to be mentioned that the Object Model of the system stores information concerning the way in which information on a fault of each particular object class is forwarded. As a consequence, the Database Process DP knows that e.g. in the case of the

2-Mbit/s interface object, fault information has to be forwarded to all objects which this interface object provides service for. However, if the fault is e.g. on one of the interfaces of a change-over unit providing protection at some level, the error is applied only to that interface (and not any further), as the system knows that this object has a back-up route, i.e. the concerned path and circuits are not out of service. In this way the basic behaviour of the system in fault conditions can be modelled inside the management system.

In the above example, each user has a Session Handler, and a separate (logical) link to the core part is arranged for each Session Handler/CAP pair. If there is more than one user on the same workstation, the same delta change or the same object has to be copied several times (once for each user), i.e. the caching of objects will be done several times. This is not the most efficient way. In particular, if the link between the Core Access Process and the Session Handler is slow, the need to send the same information twice or even more times deteriorates the performance. This drawback can be dispensed with by the embodiment shown in Figure 6, where the workstation has one common Workstation Session Handler WSH, which is connected to the different user Session Handlers SH. In this case, there is thus a single logical link between one workstation WS and the core part of the system, irrespective of the number of users of the workstation.

The peripheral part PP in the workstation comprises several user parts; the example shown in Figure 6 has two user parts UP1 and UP2 (each user part comprises the applications of the user, the Session Handler SH of the user, and the corresponding

cache CH. The caching of objects in the workstation takes place cumulatively on several different levels (here on two levels) so that, in each user Session Handler cache CH, copies of objects needed by the applications connected to that Session Handler are cached, and in the Workstation Session Handler copies of all objects needed by the applications of the workstation. The Workstation Session Handler thus knows which objects each Session Handler stores. If one of the applications needs an object which it does not currently store in its memory space, it first asks its own Session Handler whether it has the object. If this does not have a copy, it asks the Workstation Session Handler. Accordingly, the application may get the object either from another application inside the same user session or from another user's application inside the workstation session. Only in cases where not even the Workstation Session Handler has a copy of the object, the core part has to be asked for it. In this way, traffic between the workstation and the core server will be minimized, which is important particularly when the workstation and core server have been interconnected by a slow link (for instance, when they are remote from each other e.g. through a slow packet-switched network). In this embodiment, correspondingly, the table TBL2 of the core part contains information indicating which one of the Workstation Session Handlers has a copy of each particular object.

In the same way as in the first embodiment, the applications can register to the core so as to receive various events so that the core part can filter (using filter F1) events up to the application level. Alternatively, filtering can take place in steps so that filter F1 contains information indicating which

type of events each particular workstation is interested in. Then, an additional filter F2 (or a search table) must be provided in association with the Workstation Session Handler of each workstation. The filter F2 indicates on the basis of the event type, which Session Handler (i.e. which user) wants to get that particular event.

In order that the local copies and master objects could be kept consistent in the above-described system, it is to be noted that several events may simultaneously try to make changes that are inconsistent with each other. In the following, a procedure used in the system according to the invention to solve this problem will be described with reference to Figures 7 and 8 by using an application that creates circuits between objects as an example. If the user of the system wants to set up a circuit between two network elements, say A and B, he or she gives a corresponding command from his workstation. Then the application in charge of setting up circuits (Circuit Manager in figure 7) creates a link between objects A and B representing the network elements A and B. This action is recorded in the application, as shown in Figure 8. (This is what is known as action recording. It is a record of things that the user is trying to do, looking it at an application level.) This action then results in a set of changes (like set attribute X of A to be Y, set attribute X of B to be Y, set direction..., etc.) according to predetermined rules. These individual changes on individual parts of each object are then recorded, which is shown as the Changes Recording stage in Figure 8.

In a normal case, the application transmits these actions through the Session Handler to the core part, and the core part reapplies the actions to the

master objects, in order that the information of the core part on the objects would be consistent with the information of the application. The core part further transmits information on the resulting changes to all processes that have notified that they are interested in receiving such events.

When the application makes changes to the object, the core part may, however, receive information from the network to be managed indicating that there is a failure in the network element B, for instance (which may mean e.g. that the link between objects A and B cannot be created). The core part thus immediately transmit a message to the application as an indication of the failure; in the figure, the message is indicated with the reference AlarmEvAVC (Attribute Value Change event). This message carries a description of a change to the attribute value in question (it is thus a message of the same kind as the delta message in Figure 5). In this case the received information means that we now have a new alarm in network element B. The application then proceeds through the following steps:

1. All the changes that have been made at the Changes Recording stage are reversed, i.e. the application is returned to the same state where it was before the circuit set up command was given by the user.

2. Changes from outside the application are applied to the objects. In this example, this means that object B has be to moved into fault state by using the information contained in the AlarmEvAVC message.

At this stage the state of the objects (in this example object B) matches those in the core part of the system.

3. The Actions recorded earlier are reapplied one by one, and if any action fails, the application is informed. This means that the changes that the application made are retested to see whether they can still be made. This is because it is possible that the change that has occurred in the network was so insignificant that it is still possible to set up a circuit between network elements A and B.

The above-described procedure ensures that the changes made are always done on top of the core (as the changes made in the application are first taken away and then the changes that have been done in the core are added).

Even though the invention has been described above referring to the example shown in the attached drawing, it is obvious that the invention is not restricted to the example, but it can be modified in many ways within the scope of the inventive idea presented above and in the attached claims. As mentioned above, the management system includes a variety of objects, representing both logical and physical entities. In the attached claims, all these objects are set forth as "the objects that relate to the network elements to be managed", i.e. objects that represent the environment to be managed. Even though, in the typical example set forth above, the core part and peripheral parts are on different computers, so that they may be geographical very far from each other, it is equally possible to use the entire system on one and the same computer (which is typically a UNIX computer).

Claims:

1. A system for managing a telecommunications network, said telecommunications network comprising several network elements to be managed by the system,

said system comprising a management centre having at least one workstation accomplishing a man-machine interface and allowing a manager to control the system, and the system being capable of providing the manager with information on the network, said at least one workstation being connected to a database including information about the network to be managed, said information being in the form of objects that relate to the network elements to be managed and in the form of references between the different objects, said references indicating the dependencies between the objects, whereby said management centre is connected to said network elements by data communication links such that the manager can initiate an operation on and receive information from a network item to be managed, characterized in that functions of the system have been divided between a single core part and at least one peripheral part in such a way that

- of said parts, only the core part communicates with the network elements to be managed, the core part storing all objects used by the system as master copies and maintaining information concerning said objects,

- the peripheral part contains applications stored in the user's workstation, through which applications the services of the system are offered to the user, each peripheral part having separately stored copies of objects used by the applications of the concerned peripheral part and information indicating which applications use a particular object,

and

- the core part stores information indicating which peripheral part contains a copy of an individual object, whereby the peripheral parts and core parts communicate with each other to keep the objects in the core part and the copies in the peripheral parts consistent with each other.

2. A system according to claim 1, characterized in that a copy of an object is removed from the peripheral part when none of the applications needs it any longer, whereby the information indicating which applications use a particular object is updated each time an application retrieves a new object copy from the core part.

3. A system according to claim 1, characterized in that one peripheral part contains applications needed by more than one user, the peripheral part storing copies of objects cumulatively so that the objects needed by each user's applications are stored separately in their own user-specific groups, in addition to which objects needed by all applications of the peripheral part are stored in their own group.

4. A system according to claim 1, characterized in that each peripheral part is in its own workstation and each core part in its own server machine, and that there is a single link between the peripheral part and the core part for forwarding messages between the peripheral part and core part.

5. A system according to claim 1, characterized in that the core part further stores information indicating to which peripheral part different type of events from the network are to be sent.

The Patent Office

28.

| | | | |
|---|---|---|---|
| **Application No:** | GB 9526615.1 | **Examiner:** | M J Billing |
| **Claims searched:** | 1 to 6 | **Date of search:** | 30 April 1996 |

## Patents Act 1977
## Search Report under Section 17

**Databases searched:**

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A AUDB; H4K KF42.

Int Cl (Ed.6): G06F 9/46; H04Q 3/00.

Other: ONLINE : WPI.

**Documents considered to be relevant:**

| Category | Identity of document and relevant passage | Relevant to claims |
|---|---|---|
| A | EP0553560A2 (GPT) - Abstract | 1 |

| | | | |
|---|---|---|---|
| X | Document indicating lack of novelty or inventive step | A | Document indicating technological background and/or state of the art. |
| Y | Document indicating lack of inventive step if combined with one or more other documents of same category. | P | Document published on or after the declared priority date but before the filing date of this invention. |
| | | E | Patent document published on or after, but with priority date earlier than, the filing date of this application. |
| & | Member of the same patent family | | |